(finish Planning)

# Propositional Logic Intro, Syntax

## Computer Science cpsc322, Lecture 19

*(Textbook Chpt 5.1– 5.1.1 – 5.2)*

**June, 6, 2017**

# Lecture Overview

- **Recap Planning**
- Logic Intro
- Propositional Definite Clause Logic: Syntax

# Recap Planning

- Represent possible actions with ... STRIPS
- Plan can be found by... search
- Or can be found by mapping planning problem into ... CSP

# Solve planning as CSP: pseudo code

horizon = 0 ; solved = false

while not solved

map STRIPS to CSP with $\boxed{\text{horizon}}$

$\boxed{\text{solve CSP} \rightarrow \text{solution}}$

If solution found then

solved = true

else

horizon = horizon + 1

return solution

# Planning as CSP

If the algorithm for planning as CSP stops and returns a solution plan of length k, does it mean that there are no shorter solutions ?

A. Yes

there are no shorter solutions

B. No

C. It depends …

# STRIPS to CSP applet

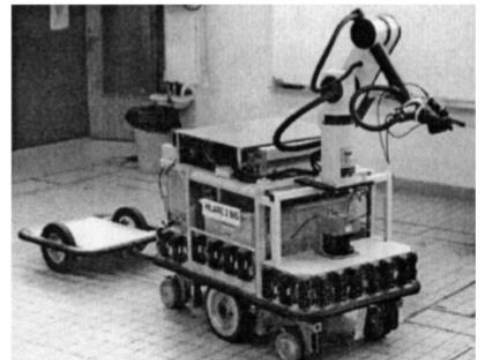
Allows you:

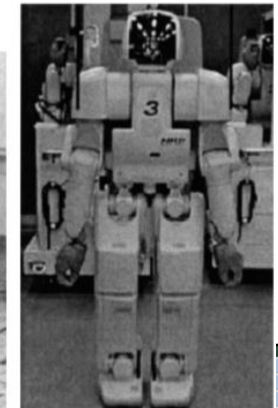
- to specify a planning problem in STRIPS
- to map it into a CSP for a given horizon
- the CSP translation is automatically loaded into the CSP applet where it can be solved

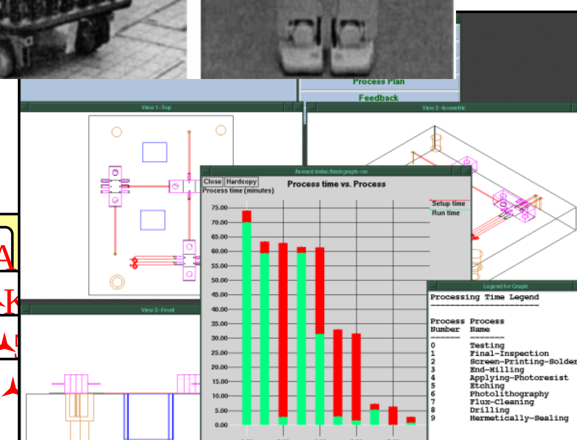
*Practice exercise using STRIPS to CSP is available on AIspace*

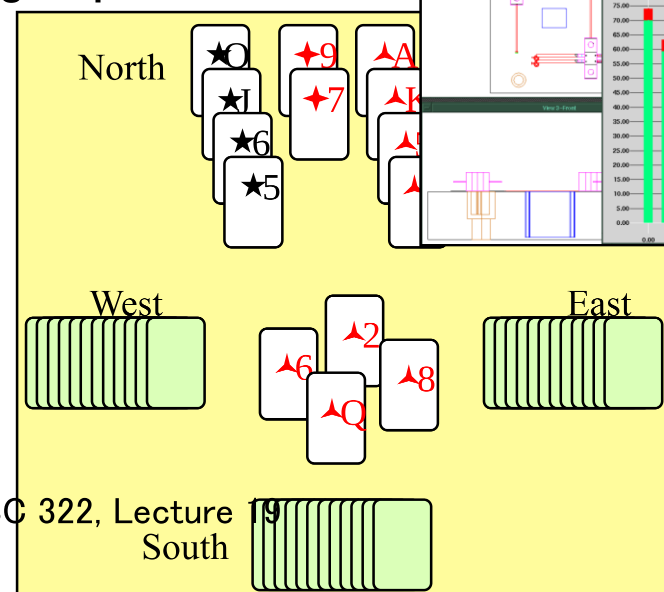
# Now, do you know how to implement a planner for....

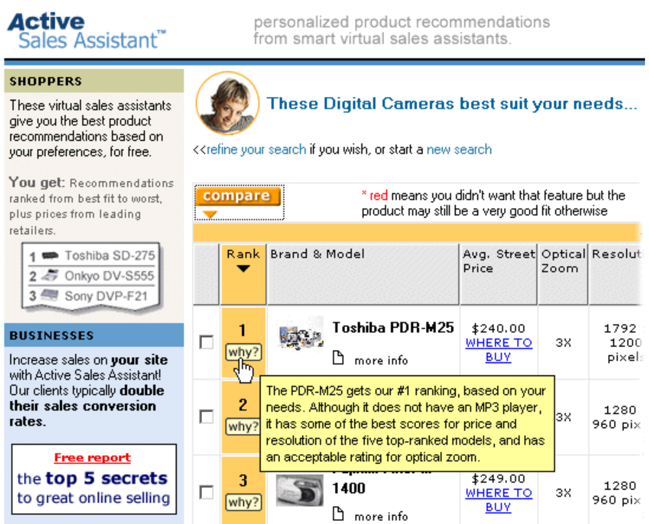
- Emergency Evacuation?
- Robotics?
- Space Exploration?
- Manufacturing Analysis?
- Games (e.g., Bridge)?
- Generating Natural language
    - Product Recommendations ….

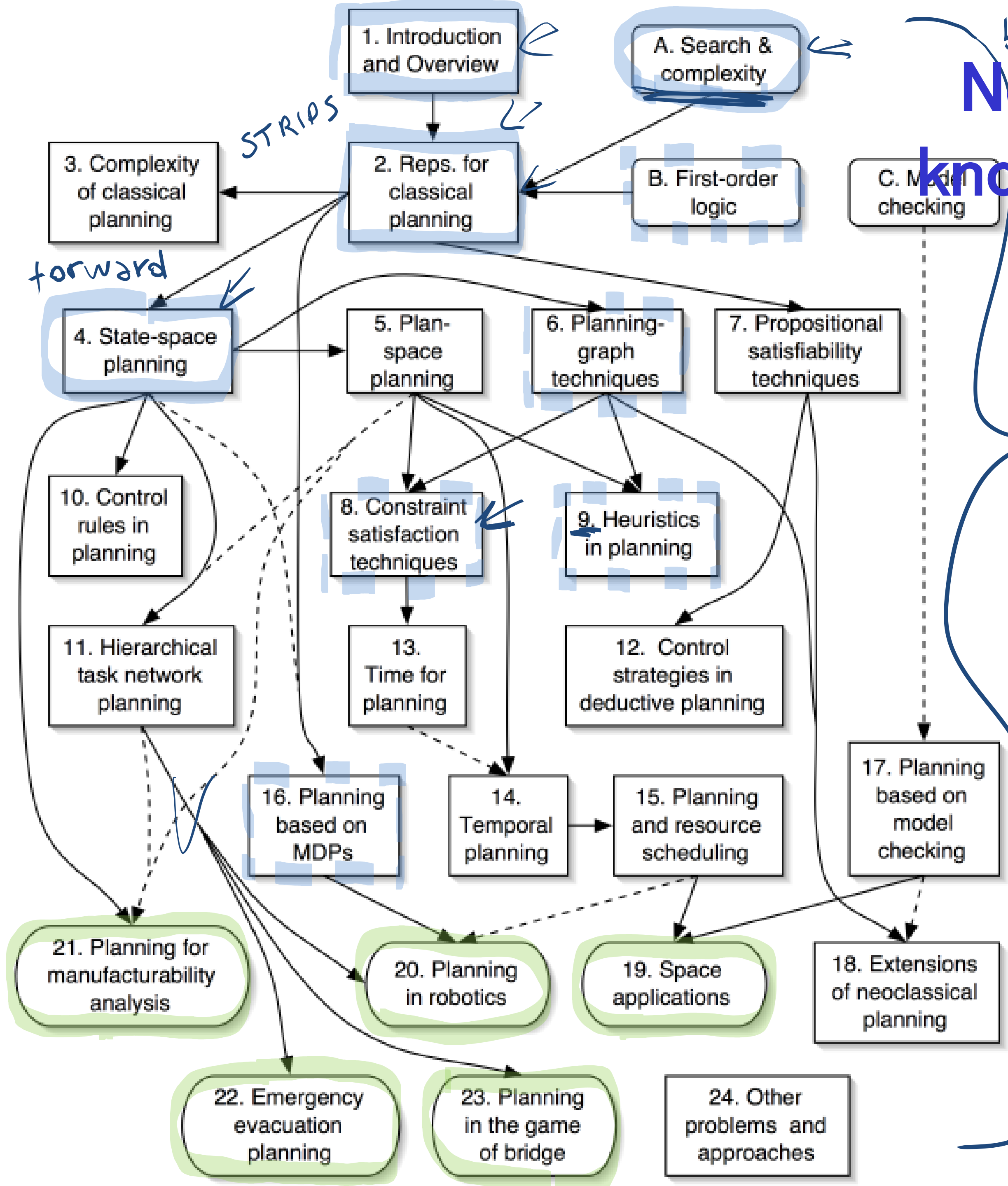









book chapters

# No ☹, but you (will) know the key ideas ☺!

Ghallab, Nau, and Traverso

***Automated Planning: Theory and Practice***

Morgan Kaufmann, May 2004

ISBN 1-55860-856-7

Web site:

- ✓ http://www.laas.fr/planning

NEW BOOK

- ✓ http://projects.laas.fr/planning/

you know

you know a little

Applications

# Lecture Overview

- Recap Planning
- **Logic Intro**
- Propositional Definite Clause Logic: Syntax

# What is coming next ?

**Environment**

| Problem | | Deterministic | Stochastic |
|---|---|---|---|
| Static | Constraint Satisfaction | *Vars + Constraints*<br>Arc Consistency<br>Search → for CSP<br>SLS | |
| | Query | CSP<br>*Logics*<br>Search → for Inference | *Belief Nets*<br>Var Elimination |
| Sequential | Planning | *STRIPS*<br>CSP<br>Search → for complex planning | *Decision Nets*<br>Var Elimination<br>*Markov Processes*<br>Value Iteration |

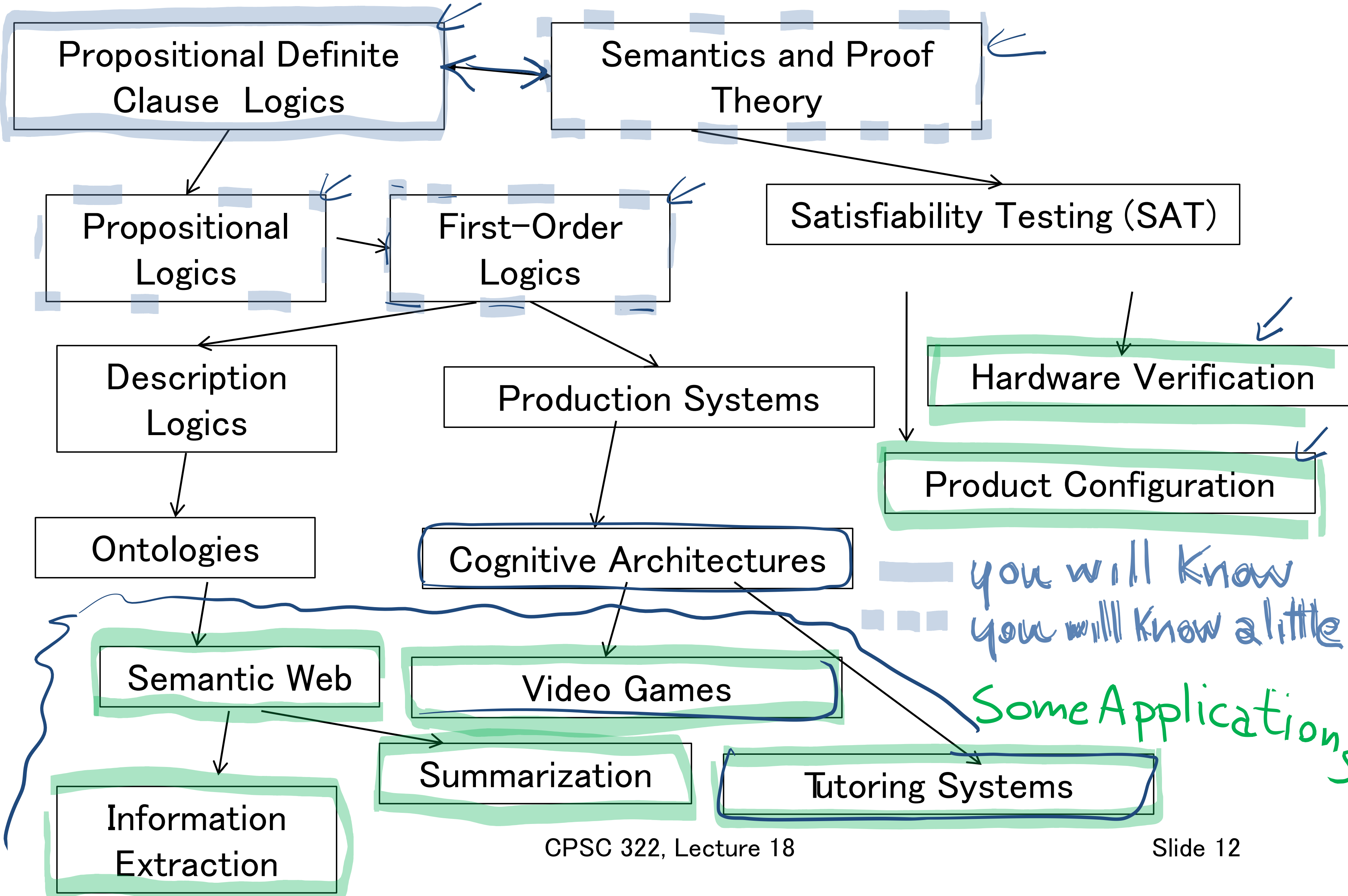
*Representation*

Reasoning Technique

# Logics

- **Mostly only propositional**…. This is the starting point for more complex ones ….
- **Natural** to express **knowledge** about the world
  - What is true (boolean variables)
  - How it works (logical formulas)
- Well understood formal properties
- Boolean nature can be exploited for efficiency
- ……

# Logics in AI: Similar slide to the one for planning



# What you already know about logic…

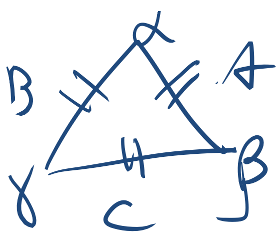
**From programming: Some logical operators**

```
If ((amount > 0) && (amount < 1000)) || !(age < 30)
...
```

AND $\wedge$ OR $\vee$ NOT $\sim, \neg$

You know what they mean in a "procedural" way

**Logic is the language of Mathematics**. To define formal structures (e.g., sets, graphs) and to proof statements about those




$\forall x \; \text{triangle}(x) \;\; A=B=C \iff \alpha=\beta=\gamma$

We are going to look at Logic as a **Representation and Reasoning System** that can be used to **formalize a domain (e.g., an electrical system, an organization)** and to **reason about it**

# Logic: A general framework for representation & reasoning

- Let's now think about **how to represent an environment** about which we have only partial (but certain) information
- What do we need to represent?

objects

events

actions

space

time

# Why Logics?

- **"Natural"** to express **knowledge** about the world

(more natural than a "flat" set of variables & constraints)

*"Every 322 student will pass the midterm"*

$Midterm(m_1)$

$Course(c_1)$

$Name\text{-}of(c_1, 322)$

$Course\text{-}of(m_1, c_1)$

$\forall z\ Student(z) \wedge Registred(z, c_1) \wedge Follows\_advice(z, Slide23)$

$\Rightarrow pass(m_1, z)$

- It is easy to incrementally add knowledge
- It is easy to check and debug knowledge
- Provide language for asking complex queries
- Well understood formal properties

# Complex Query

Student(Sue)

"will Sue pass all her midterms?"

$\forall c, m$ registred(Sue, c) $\wedge$

course-of(m, c)

? pass(m, Sue)

# Propositional Logic

We will study the simplest form of Logic: Propositional

- The primitive elements are **propositions**: Boolean variables that can be {*true, false*}

$p_1$ $p_2$

- The goal is to illustrate the basic ideas

- This is a starting point for more complex logics (e.g., first-order logic)

- Boolean nature can be exploited for efficiency.

# Propositional logic: Complete Language

The **proposition** symbols $p_1$, $p_2$ ··· etc are sentences

- If S is a sentence, $\neg$**S** is a sentence (negation)
- If $S_1$ and $S_2$ are sentences, $\mathbf{S_1 \wedge S_2}$ is a sentence (conjunction)
- If $S_1$ and $S_2$ are sentences, $\mathbf{S_1 \vee S_2}$ is a sentence (disjunction)
- If $S_1$ and $S_2$ are sentences, $\mathbf{S_1 \Rightarrow S_2}$ is a sentence (implication)
- If $S_1$ and $S_2$ are sentences, $\mathbf{S_1 \Leftrightarrow S_2}$ is a sentence (biconditional)

Sample Formula

$$((p_1 \vee p_2) \wedge p_3) \Leftrightarrow ((p_2 \Rightarrow \neg p_4) \vee p_5)$$

# Propositional Logics in practice

- Agent is told (perceives) some facts about the world
  some propositions are true

- Agent is told (already knows / learns) how the world works
  logical formulas

- Agent can answer yes/no questions about whether other facts must be true

# Using Logics to make inferences…

1) Begin with a **task domain**.

2) Distinguish those **things** you want to talk about (the ontology).

SEE NEXT SLIDE

3) Choose symbols in the computer to **denote propositions**

live_$w_6$ sw$_1$_on

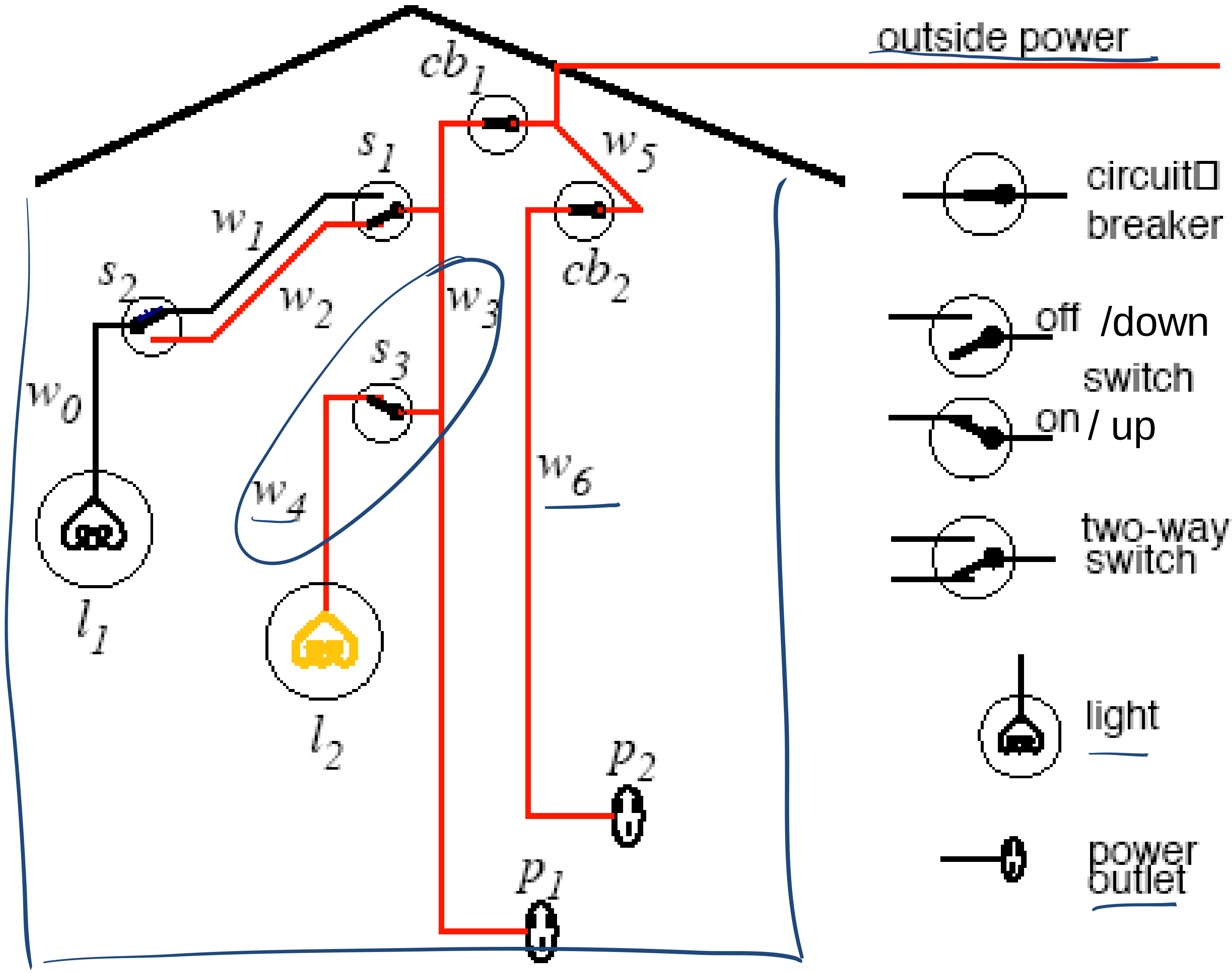
4) Tell the system **knowledge** about the domain.

live_$w_3$ $\wedge$ sw$_3$_on $\rightarrow$ live_$w_4$

5) **Ask the system** whether new statements about the domain are true or false.

$l_2$_on ?

# Electrical Environment



# Lecture Overview

- Recap Planning
- Logic Intro
- **Propositional Definite Clause Logic: Syntax**

# Propositional Definite Clauses

- **Propositional Definite Clauses:** our first logical representation and reasoning system.

(very simple!)

- Only two kinds of statements:
  - that **a proposition is true**
  - that **a proposition is true if one or more other propositions are true**

$p_1 \quad p_2$

$p_1 \leftarrow p_3 \wedge p_4$

- Why still useful?
  - Adequate in many domains (with some adjustments)
  - Reasoning steps easy to follow by humans
  - Inference linear in size of your set of statements
  - Similar formalisms used in cognitive architectures

# Propositional Definite Clauses: Syntax

**Definition (atom)**

An **atom** is a symbol starting with a lower case letter

$p_1$

**Definition (body)**

$p_1 \wedge \ldots \wedge p_n$

A **body** is an atom or is of the form $b_1 \wedge b_2$ where $b_1$ and $b_2$ are bodies.

**Definition (definite clause)**

$p_1 \leftarrow \underbrace{p_2 \wedge . p_5}$

A **definite clause** is an atom or is a rule of the form $\boxed{h \leftarrow b}$ where $h$ is an atom and $b$ is a body. (Read this as ``$h$ if $b$.'')

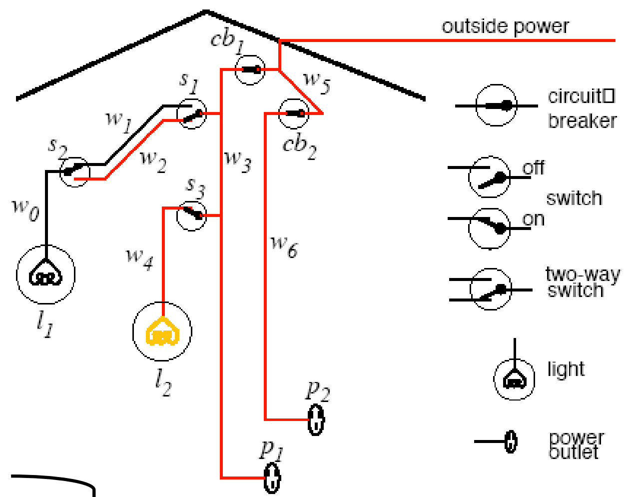
**Definition (KB)**

A **knowledge base** is a set of definite clauses

clause$_1$
⋮
clause$_n$

$light\_l_1.$
$light\_l_2.$
$ok\_l_1.$
$ok\_l_2.$
$ok\_cb_1.$
$ok\_cb_2.$
$live\_outside.$

atoms




definite clauses, KB

$live\_l_1 \leftarrow live\_w_0.$
$live\_w_0 \leftarrow live\_w_1 \wedge up\_s_2.$
$live\_w_0 \leftarrow live\_w_2 \wedge down\_s_2.$
$live\_w_1 \leftarrow live\_w_3 \wedge up\_s_1.$
$live\_w_2 \leftarrow live\_w_3 \wedge down\_s_1.$
$live\_l_2 \leftarrow live\_w_4.$
$live\_w_4 \leftarrow live\_w_3 \wedge up\_s_3.$
$live\_p_1 \leftarrow live\_w_3.$
$live\_w_3 \leftarrow live\_w_5 \wedge ok\_cb_1.$
$live\_p_2 \leftarrow live\_w_6.$
$live\_w_6 \leftarrow live\_w_5 \wedge ok\_cb_2.$
$live\_w_5 \leftarrow live\_outside.$
$lit\_l_1 \leftarrow light\_l_1 \wedge live\_l_1 \wedge ok\_l_1.$
$lit\_l_2 \leftarrow light\_l_2 \wedge live\_l_2 \wedge ok\_l_2.$

rules

# PDC Syntax: more examples

**Definition (definite clause)**

A **definite clause** is

- an atom or
- a rule of the form $h \leftarrow b$ where $h$ is an atom ('head') and $b$ is a body. (Read this as '$h$ if $b$.')

iclicker.

A. Legal B. Not Legal

a) *ai_is_fun*

b) *ai_is_fun* $\vee$ *ai_is_boring*

c) *ai_is_fun* $\leftarrow$ *learn_useful_techniques*

d) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge$ *notTooMuch_work*

e) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge \neg$ *TooMuch_work*

f) *ai_is_fun* $\leftarrow$ *f(time_spent, material_learned)*

g) *srtsyj* $\leftarrow$ *errt* $\wedge$ *gffdgdgd*

# PDC Syntax: more examples

Legal PDC clause | Not a legal PDC clause

a) *ai_is_fun* (Legal)

b) *ai_is_fun* $\vee$ *ai_is_boring* (Not legal)

c) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* (Legal)

d) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge$ *notTooMuch_work* (Legal)

e) *ai_is_fun* $\leftarrow$ *learn_useful_techniques* $\wedge$ $\neg$ *TooMuch_work* (Not legal)

f) *ai_is_fun* $\leftarrow$ *f(time_spent, material_learned)* (Not legal)

g) *srtsyj* $\leftarrow$ *errt* $\wedge$ *gffdgdgd* (Legal)

Do any of these statements mean anything?
Syntax doesn't answer this question!

# Learning Goals for today's class

**You can:**

- Verify whether a logical statement belongs to the language of full propositional logics.
- Verify whether a logical statement belongs to the language of propositional definite clauses.

# Study for midterm (Thurs June 8)

Midterm: ~6 short questions (*8pts each*) + 2 problems (*26pts each)*

- Study: textbook and **inked** slides
- Work on **all** practice exercises and **revise assignments**!
- While you revise the **learning goals**, work on **review questions** – I may even reuse some verbatim ☺
- Work on the **couple of problems** from previous offering (maybe slightly more difficult) … but you have the solutions ☺

# Next class

- Definite clauses Semantics and Proofs (textbook 5.1.2, 5.2.2)